Um Método para Avaliação de Resiliência de Firmwares IoT por Injeção Sistemática de Falhas Sensoriais Compostas

Cleber S. Peter[1], Alexandre R. R. de Souza[2], Hélida Santos[3], Giancarlo Lucca[3], Renata Reiser[1], Adenauer C. Yamin[1]

[1]Univ. Federal de Pelotas (UFPel)
[2]Inst. Federal do Rio Grande do Sul (IFRS)
[3]Univ. Federal do Rio Grande (FURG)

{cdspeter, reiser, adenauer}@inf.ufpel.edu.br,
alexandre.souza@riogrande.ifrs.edu.br,
{helidasantos, giancarlolucca}@furg.br

Abstract. *IoT nodes deployed in field operations are susceptible to progressive sensory degradation, yet no systematic method exists to verify whether embedded firmware can withstand such conditions before deployment. This paper proposes a deterministic co-simulation method to assess firmware resilience through the systematic injection of single and compound faults. The method integrates a five-mode fault taxonomy with dependability metrics to reliably assess the underlying system. Applied to the Golioth Air Quality Monitor, the method successfully discriminated detectable from undetectable fault modes, demonstrating its effectiveness in assessing firmware resilience before field deployment.*

Resumo. *Nós IoT operando em campo encontram-se suscetíveis a processos progressivos de degradação sensorial, contudo inexistem métodos sistemáticos para verificar se o firmware embarcado é capaz de tolerá-los antes da implantação. Este artigo propõe um método de co-simulação determinística para avaliar a resiliência de firmwares reais por meio da injeção sistemática de falhas unitárias e compostas. A avaliação integra uma taxonomia de cinco modos de falha com métricas de dependabilidade para avaliar a resiliência dos nós de forma sistemática. Aplicado ao Golioth Air Quality Monitor, o método demonstrou sua eficácia ao distinguir com precisão os modos de falha detectáveis dos indetectáveis, validando sua capacidade de verificar a resiliência de firmwares antes da implantação em campo.*

1. Introdução

A computação ubíqua e pervasiva concretiza-se, em grande parte, por meio de redes de nós sensores distribuídos em ambientes inteligentes, como edifícios, plantações agrícolas, data centers e espaços urbanos, que operam de forma autônoma e prolongada, coletando grandezas físicas e publicando dados para plataformas de análise. Além disso, o paradigma da Internet das Coisas (IoT) pressupõe que esses dispositivos sejam *deploy-and-forget*, ou seja, uma vez instalados devem produzir dados confiáveis com mínima intervenção humana [Firouzi et al. 2022].

Contudo, a operação prolongada em campo expõe os sensores a processos degenerativos que comprometem progressivamente a qualidade das medições. Deriva térmica, condensação, acumulação de partículas, envelhecimento eletroquímico e interferência

eletromagnética constituem modos de falha comuns que podem manifestar-se como desvios graduais (*drift*), valores travados (*stuck-at*), picos espúrios (*spikes*), deslocamentos abruptos (*offset*) ou até mesmo a perda total do sinal (*blackout*) [Gaddam et al. 2020]. Quando não detectadas, essas anomalias propagam-se como dados corrompidos através de toda a cadeia de processamento, levando a falsos alertas, decisões equivocadas e contribuem para a perda de confiança no sistema [Erhan et al. 2021].

Para mitigar esses efeitos, o firmware embarcado dos nós IoT frequentemente incorpora mecanismos de resiliência tais como filtros de mediana, detecção estatística de *outliers*, *watchdogs* de plausibilidade e estratégias de *fallback* para o último valor válido [He et al. 2022]. No entanto, cada técnica é eficaz apenas contra um subconjunto restrito de modos de falha. Filtros de mediana suprimem picos espúrios e falham na detecção de derivas graduais, ao passo que *watchdogs* de plausibilidade identificam valores anômalos sem perceber deslocamentos contidos nos limites aceitos [Gaddam et al. 2020, Erhan et al. 2021]. Essa seletividade se agrava quando múltiplos sensores interagem na lógica de aplicação, visto que a combinação de modos de falha distintos pode anular ou mascarar mecanismos de proteção que seriam eficazes isoladamente [He et al. 2022].

Apesar dessa dependência, a validação desses mecanismos na fase de desenvolvimento continua sendo uma lacuna metodológica crítica. Testes unitários convencionais operam quase exclusivamente sobre *mocks* com valores constantes, mostrando-se estruturalmente incapazes de reproduzir as dinâmicas temporais de degradação contínua [Birchler et al. 2025]. Os ensaios em campo, por sua vez, são onerosos, não reprodutíveis e limitam o controle sistemático das variáveis físicas. Esse distanciamento faz com que vulnerabilidades arquiteturais sejam frequentemente descobertas apenas após a implantação física.

Diante desse cenário, este artigo tem como objetivo apresentar um método para avaliação de resiliência de firmwares IoT por injeção sistemática de falhas sensoriais progressivas e compostas, baseado em co-simulação heterogênea. O método acopla o firmware a um modelo ambiental e aplica degradações parametrizáveis de forma sistemática sob condições controladas. A principal contribuição reside na formalização de uma abordagem de validação de resiliência sistêmica, demonstrada experimentalmente com o *Golioth Air Quality Monitor* [Golioth, Inc. 2026]. Os resultados evidenciam vulnerabilidades nos mecanismos de proteção do *firmware* que permanecem latentes nos testes unitários convencionais, confirmando a capacidade do método de revelar o impacto de falhas compostas.

Para fundamentar o método, a Seção 2 apresenta a fundamentação conceitual, e a Seção 3 formaliza o método concebido, incluindo a taxonomia de falhas e as métricas de resiliência. A Seção 4 apresenta o estudo de caso e as avaliações realizadas. A Seção 5 situa os resultados obtidos no contexto das linhas de investigação correlatas e, por fim, a Seção 6 delineia as perspectivas decorrentes.

2. Fundamentação Conceitual

A classificação sistemática dos modos de falha sensoriais constitui o alicerce terminológico do presente trabalho. [Ni et al. 2009] foram pioneiros ao propor uma taxonomia formal que unificou a nomenclatura dispersa na literatura, estabelecendo critérios objetivos de diferenciação baseados no perfil temporal e na amplitude de cada perturbação.

Essa taxonomia consolidou categorias que até então eram tratadas de forma *ad hoc*, permitindo que trabalhos subsequentes adotassem um vocabulário técnico comum. [Gaddam et al. 2020] ampliaram esse arcabouço ao formalizar as fronteiras conceituais entre falha, anomalia e *outlier* no domínio IoT, demonstrando que cada categoria possui dinâmicas temporais e perfis estatísticos próprios. Essa distinção é relevante porque impõe requisitos diferenciados aos mecanismos de defesa, que necessitam ser projetados e avaliados separadamente para cada classe de perturbação.

A literatura sobre defesas embarcadas contra falhas sensoriais é ampla e fragmentada. [Erhan et al. 2021] conduziram a revisão mais abrangente até o momento, catalogando técnicas que vão de limiares estatísticos a redes neurais profundas e concluíram que a eficácia de cada método é intrinsecamente condicionada ao modo de falha alvo. Esse resultado reforça a necessidade de avaliações que exercitem simultaneamente múltiplos modos de perturbação. Complementarmente, [He et al. 2022] demonstraram que a composição colaborativa de camadas heterogêneas de filtragem e validação eleva a robustez global do sistema, consolidando a premissa de que defesas multicamada constituem o estado da prática em *firmware* IoT.

A co-simulação heterogênea consiste na orquestração sincronizada de múltiplos simuladores dotados de semânticas computacionais independentes, permitindo avaliar o comportamento global de um sistema ciber-físico antes da sua implantação [Birchler et al. 2025]. Essa técnica materializa o paradigma denominado *shift-left testing*, que preconiza antecipar a detecção de defeitos para as fases iniciais do ciclo de desenvolvimento. [Minerva et al. 2020] destacam que a fidelidade do modelo ambiental e a sincronização determinística entre os domínios são requisitos indispensáveis para que os resultados obtidos em simulação sejam representativos do comportamento real. [Arrieta et al. 2019] complementam essa visão ao demonstrar que a co-simulação viabiliza a exploração automatizada de grandes espaços de configuração em testes de sistemas complexos, ampliando a cobertura alcançável em campanhas experimentais.

A avaliação formal da capacidade de um sistema tolerar e isolar perturbações repousa sobre o conceito de dependabilidade definido por [Avizienis et al. 2004], que descreve a cadeia causal na qual uma falta se manifesta como erro e pode propagar-se até uma falha observável. A injeção de falhas implementada por *software*, denominada SWIFI, constitui o método experimental canônico para percorrer essa cadeia de forma controlada. [Natella et al. 2016] apresentam a revisão mais abrangente sobre essas técnicas, sistematizando como perturbações deliberadas permitem mensurar a cobertura das defesas ativas de um sistema. Mais recentemente, [Cotroneo et al. 2022] avançam com uma abordagem analítica que correlaciona padrões de injeção a modos de falha latentes, ampliando o potencial diagnóstico da técnica.

A convergência desses quatro pilares delineia o espaço conceitual que sustenta o presente trabalho. A taxonomia de falhas sensoriais define o catálogo de perturbações a serem reproduzidas experimentalmente. Os mecanismos de detecção e defesa constituem o objeto sob avaliação. A co-simulação heterogênea fornece a infraestrutura de execução determinística para acoplar o *firmware* real a modelos ambientais. A injeção por *software* estabelece o protocolo de perturbação controlada. Essa articulação conduz ao método concebido, especificado na seção seguinte.

3. Método Concebido para Injeção Sistemática de Falhas Sensoriais

A Figura 1 apresenta a arquitetura do ambiente de avaliação, estruturada em dois domínios acoplados. O **domínio de simulação** reúne o modelo ambiental, responsável por gerar sinais contínuos para cada grandeza física monitorada, e o motor de injeção de falhas, que aplica perturbações parametrizadas sobre eles. O **domínio embarcado virtualizado** executa o firmware IoT como processo nativo de software sem modificações no código-fonte. Para isso, a camada de *Hardware Abstraction Layer* (HAL) virtual substitui o acesso direto aos periféricos físicos pelos sinais provenientes do domínio de simulação e preserva a interface original do firmware. Enfim, um monitor de co-simulação observa ambos os domínios e centraliza os registros de cada passo da campanha.

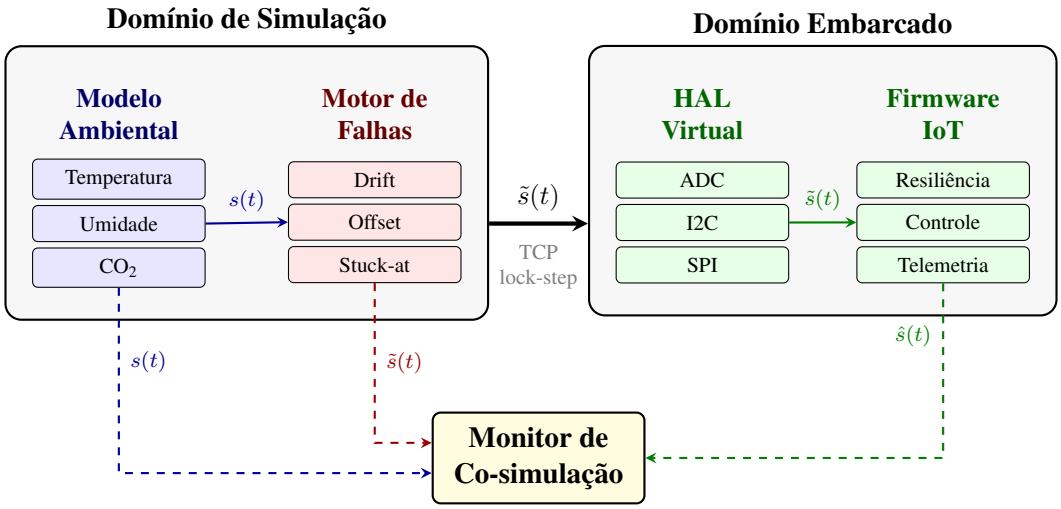


Figura 1. Arquitetura do ambiente de avaliação de resiliência.

3.1. Visão Geral do Método

A cada passo Δt, o modelo ambiental produz o sinal de referência $s(t)$. O motor de injeção de falhas aplica a transformação associada ao modo ativo e gera o sinal corrompido $\tilde{s}(t)$, que é transmitido via rede ao domínio embarcado. A HAL virtual entrega $\tilde{s}(t)$ ao firmware com a mesma assinatura da interface física. Em seguida, o firmware processa a leitura, executa seus mecanismos de resiliência e publica o valor tratado $\hat{s}(t)$. O monitor de co-simulação registra os três sinais e implementa a sincronização denominada *lock-step*, na qual o passo seguinte avança apenas após a confirmação de conclusão do passo atual por ambos os domínios. Esse mecanismo é fundamental para garantir a sincronização determinística da co-simulação e a reprodutibilidade dos testes independentemente das variações no tempo de processamento da máquina hospedeira.

Todo esse ciclo viabiliza uma avaliação controlada da resiliência ao permitir que o desenvolvedor configure o cenário de falha enquanto a co-simulação reproduz dias de operação em poucos segundos. Ao término da campanha, os registros centralizados são processados para o cálculo das métricas definidas na Seção 3.3, quantificando objetivamente a capacidade do firmware em detectar cada modo de degradação. Por executar o software embarcado integralmente sem qualquer modificação em seu código-fonte original, os resultados alcançados refletem fielmente o real comportamento do dispositivo físico alvo em campo.

3.2. Taxonomia de Modos de Falha

Cinco modos de falha são definidos para modelar os principais mecanismos de degradação observados em sensores IoT [Ni et al. 2009, Gaddam et al. 2020]. A Tabela 1 sintetiza cada modo e seu modelo matemático correspondente.

Tabela 1. Taxonomia de modos de falha de sensores.

Modo de falha	**Modelo $\tilde{s}(t)$**	**Parâm.**	**Descrição**
Deriva (*drift*)	$s(t) + \alpha(t - t_f)$	α, t_f	Acumulação monótona
Travamento (*stuck-at*)	$s(t_f)$	t_f	Congela no último valor
Pico (*spike*)	$s(t) + A\,\delta(t - t_k)$	A, t_k	Impulso espúrio
Deslocamento (*offset*)	$s(t) + \Delta,\ t \geq t_f$	Δ, t_f	Desvio constante
Apagão (*blackout*)	$0,\ t \geq t_f$	t_f	Perda total do sinal

Os parâmetros de cada modo são configuráveis por cenário de teste, permitindo variar tanto a severidade quanto o instante de ativação. Além dos modos unitários, a metodologia suporta cenários compostos nos quais múltiplos modos atuam simultaneamente sobre sensores distintos, exercitando a interação entre mecanismos de resiliência.

3.3. Métricas de Resiliência

Para quantificar o comportamento do firmware frente a cada cenário de falha, adotam-se cinco métricas complementares, derivadas de conceitos clássicos de dependabilidade [Avizienis et al. 2004] e diagnóstico de falhas [Isermann 2005]. Sendo N o número total de amostras coletadas e N_f o número de amostras no período de falha ativa, a Tabela 2 sintetiza as definições adotadas.

Tabela 2. Métricas de resiliência.

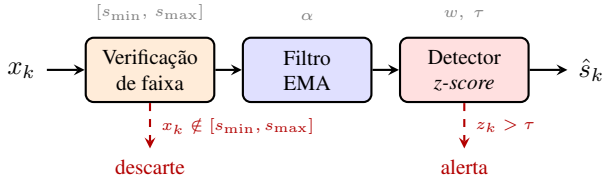
Métrica	**Definição**	**Descrição**
Tempo de detecção (T_d)	$t_{\text{det}} - t_f$	Intervalo até a sinalização
Qualidade (Q)	$\sqrt{\frac{1}{N_f}\sum(s_i - \hat{s}_i)^2}$	RMSE sob falha
Taxa de detecção (P_d)	VP/N_f	Sensibilidade
Falsos alarmes (P_{fa})	$\text{FP}/(N - N_f)$	Falsos positivos
Disponibilidade (A)	$\frac{1}{N}\sum \mathbf{1}[\lvert s_i - \hat{s}_i\rvert < \epsilon]$	Fração dentro de ϵ

Essas métricas permitem comparar objetivamente diferentes configurações de firmware, identificando tanto a capacidade de detecção quanto o impacto residual de cada modo de falha sobre a qualidade dos dados publicados.

4. Estudo de Caso

Embora o método proposto seja genérico e reutilizável em diferentes arquiteturas IoT, esta seção o instancia no *Golioth Air Quality Monitor* para demonstrar sua viabilidade prática. Esse dispositivo alvo consiste em um *design* de referência *open-source* construído sobre o Zephyr RTOS [Zephyr Project 2026], que integra sensores BME280 para aquisição de temperatura e umidade, SCD4x para CO_2 e SPS30 para material particulado. Essa combinação cobre grandezas com dinâmicas temporais distintas, tornando a plataforma representativa de dispositivos de borda em cenários de monitoramento ambiental.

A Figura 2 detalha o *pipeline* de validação e os parâmetros implementado no firmware. Cada leitura x_k é primeiro submetida a uma verificação de faixa $[s_{\min},\, s_{\max}]$ baseada no *datasheet* do sensor e, em seguida, suavizada pelo filtro EMA $y_k = \alpha\, x_k + (1-\alpha)\, y_{k-1}$, que atenua picos de curta duração. O sinal filtrado alimenta então o detector de *z-score* móvel $z_k = |x_k - \bar{x}_w| \,/\, \sigma_w$, que compara cada amostra com a dispersão da janela de w amostras anteriores e emite um alerta sempre que z_k excede o limiar τ.



Canal	$[s_{\min}, s_{\max}]$	α	w	τ
Temp. (°C)	$[-40, 85]$	0,05	120	3,5
Umid. (%)	$[0, 100]$	0,10	100	3,5
CO_2 (ppm)	$[400, 5000]$	0,20	60	4,0

Figura 2. *Pipeline* de validação do firmware (esq.) e parâmetros por canal (dir.).

O modelo ambiental em Scilab/Xcos [Scilab Enterprises 2026] reproduz ciclos de 24 h com $\Delta t = 1$ s e falhas ativadas em $t_f = 8$ h. Os sinais são senoidais e com ruído gaussiano ($\sigma = 0{,}5\,\%$ do fundo de escala), valor escolhido por manter variação de baixa amplitude frente ao sinal útil sem saturar os filtros. A título de exemplo, a Figura 3 apresenta a matriz de modos de falha aplicados ao canal de temperatura. A primeira linha exibe os modos unitários de deriva, deslocamento e travamento, enquanto a segunda acrescenta pico e apagão e introduz a primeira combinação, de deriva com deslocamento. A terceira linha reúne exclusivamente falhas compostas envolvendo dois ou três modos simultâneos.

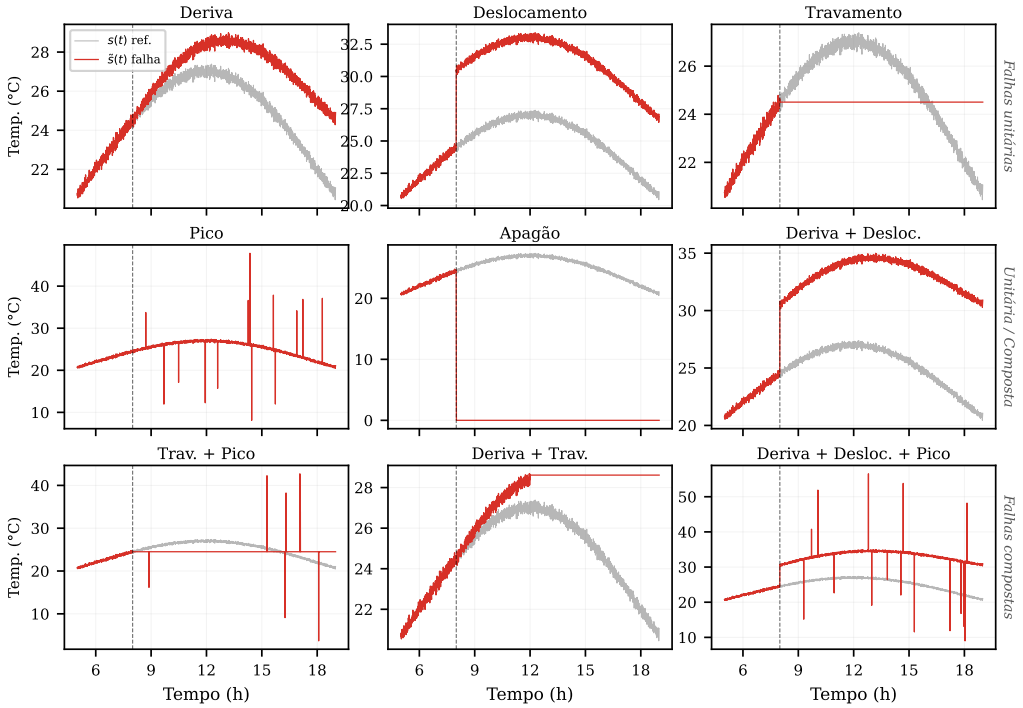


Figura 3. Matriz de falhas sobre o canal de temperatura. Cinza: $s(t)$ referência; vermelho: $\tilde{s}(t)$ corrompido; tracejada: $t_f = 8$ h.

O *firmware* alvo utiliza o ecossistema do Zephyr RTOS compilado com o *target* `native_sim`. O Zephyr foi selecionado por prover a (`native_sim`) como uma plataforma de virtualização robusta que transpõe a execução do sistema operacional diretamente para o espaço do hospedeiro nativo Linux mantendo, assim, inalterados os recursos da aplicação em teste.

A validação dos dados publicados é realizada por meio de um *dashboard* Grafana que consome a telemetria via InfluxDB. A escolha pela manutenção dos dados em um ambiente local, em vez de utilizar uma solução em nuvem, foi motivada pela necessidade de isolamento da aplicação e garantia da avaliação dos mecanismos de detecção na borda sem interferência de latências ou oscilações inerentes a conexões em nuvem. O *dashboard* reproduz a visão que um operador teria em produção, exibindo as séries temporais de cada canal, o *status* de alertas gerados pelo detector *z-score* e a comparação entre o sinal publicado e a referência do modelo ambiental. O cálculo das métricas (T_d, Q, P_d, P_{fa} e A) é realizado automaticamente no pós-processamento dos registros do monitor de co-simulação. Desse modo, o ambiente de avaliação integra a co-simulação à cadeia de observabilidade que o firmware utilizaria em campo.

4.1. Avaliações Realizadas

A campanha totaliza 23 execuções de 24 h cada, sendo 3 nominais, 15 unitárias cobrindo cinco modos aplicados a três canais e 5 compostas com falhas simultâneas em canais distintos. Os parâmetros de injeção são normalizados pelo fundo de escala de cada canal para garantir severidade equivalente. A Tabela 3 apresenta a disponibilidade A por canal, a qualidade Q como RMSE normalizado pelo fundo de escala e a taxa de detecção P_d, definida como a fração de componentes de falha detectados. Para os cenários unitários, apenas o canal injetado é reportado, pois os demais canais mantiveram $A = 1{,}00$.

Tabela 3. Resultados da campanha ($P_{fa} < 10^{-4}$ em todos os cenários).

Cenário	**Disponibilidade (A)**			Q	T_d	P_d	**Detecção**
	Temp.	Umid.	CO_2				
Nominal ($\times 3$)	1,00	1,00	1,00	—	—	—	—
Deslocamento ($\times 3$)	0,33	0,33	0,33	0,16	<1 s	1,00	Total
Apagão ($\times 3$)	0,33	0,33	0,33	0,40	<1 s	1,00	Total
Pico ($\times 3$)	1,00	1,00	1,00	0,01	<1 s	1,00	Total
Travamento ($\times 3$)	0,42	0,45	0,36	0,07	∞	0,00	Nenhuma
Deriva ($\times 3$)	0,83	0,87	0,78	0,04	∞	0,00	Nenhuma
C_1: drv_T + dsl_U	0,87	0,65	1,00	0,04	∞	0,00	Nenhuma
C_2: trv_T + drv_C	0,42	1,00	0,80	0,07	∞	0,00	Nenhuma
C_3: dsl_T + apg_U	0,33	0,33	1,00	0,40	<1 s	1,00	Total
C_4: pco_T + trv_C	1,00	1,00	0,36	0,07	<1 s	0,50	Parcial
C_5: drv_T + dsl_U + pco_C	0,83	0,33	1,00	0,16	<1 s	0,67	Parcial

Abrev.: drv = deriva, dsl = deslocamento, trv = travamento, pco = pico, apg = apagão.

Subíndices: T = temperatura, U = umidade, C = CO_2. Parcial: ao menos um componente não detectado.

Os cenários unitários dividem-se em dois grupos. Deslocamento, apagão e pico geram descontinuidades detectadas em menos de um segundo, com $P_d = 1{,}00$. O apagão apresenta o maior erro agregado, $Q = 0{,}40$, por eliminar completamente o sinal durante todo o período de falha, enquanto o pico afeta uma única amostra com $Q = 0{,}01$ e preserva $A = 1{,}00$. O segundo grupo, formado por travamento e deriva, não gera qualquer alerta, ambos com $P_d = 0{,}00$. A degradação gradual mantém A entre $0{,}78$ e $0{,}87$, mascarando a corrupção sob valores de disponibilidade aparentemente aceitáveis.

Os cenários compostos revelam três regimes. Quando ambos os modos são graduais, como em C_1 e C_2, nenhum alerta é emitido e $P_d = 0{,}00$. Quando ambos são abruptos, como em C_3, a detecção é completa com $P_d = 1{,}00$. Os cenários mistos expõem detecção parcial. Em C_4, o pico em temperatura é detectado, mas o travamento em CO_2 permanece invisível, resultando em $P_d = 0{,}50$. Em C_5, dois dos três componentes são detectados com $P_d = 0{,}67$, enquanto a deriva em temperatura prossegue silenciosamente.

Como verificação de estabilidade, os cenários críticos também foram executados com duas sementes independentes (A/B). Nas métricas de qualidade e disponibilidade observou-se baixa variação entre sementes (coeficiente de variação abaixo de 2,3%), enquanto P_d nos modos graduais apresentou maior sensibilidade ao ruído e aos limiares, compatível com sinais próximos do limiar de decisão.

4.2. Resposta a Falhas Abruptas

A Figura 4 detalha o deslocamento de $+8\,°\text{C}$ em temperatura em $t_f = 8\,\text{h}$, representativo dos modos detectáveis. Os painéis à esquerda cobrem as 24 h completas e os painéis à direita ampliam a transição em torno de t_f, evidenciando a divergência entre $\hat{s}(t)$ e $s(t)$ e a resposta do detector com indicação binária de alerta.

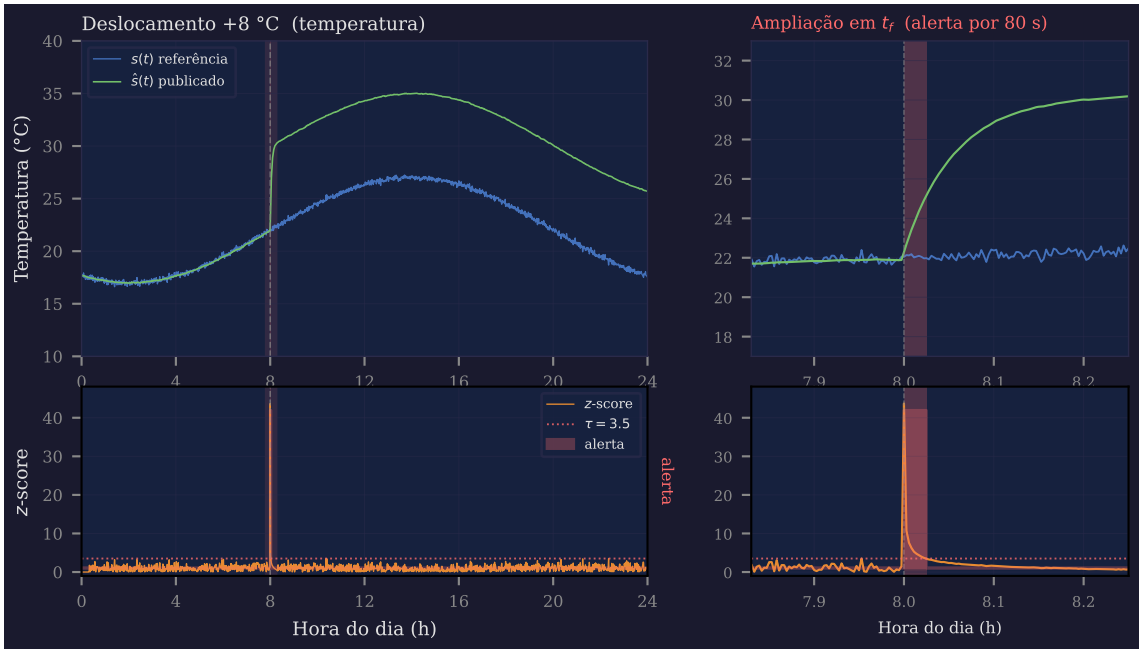


Figura 4. Deslocamento de $+8\,°\text{C}$ em temperatura, visão de 24 h e ampliação em t_f com sinal de alerta.

A detecção decorre da interação entre os dois estágios do *pipeline*. O filtro EMA, com $\alpha = 0{,}05$, não absorve descontinuidades instantâneas, propagando o degrau inte-

gralmente ao detector de *z-score*, cuja janela de $w = 120$ amostras ainda contém leituras do regime nominal. O *z-score* atinge $z = 43{,}7$ logo após t_f e permanece acima de τ por 80 s, correspondendo a nove alertas consecutivos visíveis na ampliação, até que a janela absorva o novo patamar. O $T_d < 1$ s observado uniformemente em deslocamento, apagão e pico na Tabela 3 confirma que qualquer descontinuidade com magnitude superior a $\tau \cdot \sigma_w$ é capturada independentemente do canal ou do modo. Os demais canais mantiveram $A = 1{,}00$, evidenciando o isolamento por canal.

Esse resultado evidencia que o *pipeline* oferece proteção eficaz contra variações abruptas. A co-simulação permite validar esse comportamento em condições realistas, reproduzindo a propagação completa do sinal desde a injeção de falha até a emissão do alerta pelo firmware sobre `native_sim`.

4.3. Resposta a Deriva Composta

O cenário C_1 combina deriva de $+0{,}20\,°$C/h em temperatura e deslocamento gradual de $+4\,\%$ em umidade, ambos ativados em $t_f = 8$ h. Ao final das 16 h restantes sob falha, a temperatura acumula $+3{,}2\,°$C de desvio e a umidade estabiliza com $+4\,\%$ após 2 h. Apesar da corrupção progressiva visível nas séries temporais, nenhum alerta é emitido em qualquer canal durante toda a simulação.

A evasão decorre da natureza adaptativa do detector. A janela de w amostras recalcula $\bar{x}_w$ e σ_w a cada passo, de modo que variações cuja taxa por amostra permanece inferior a τ são absorvidas pela própria estatística de referência. Em temperatura, a deriva de $+0{,}20\,°$C/h corresponde a $+0{,}002\,°$C por amostra, fração desprezível frente ao desvio padrão da janela. Em umidade, o deslocamento é distribuído ao longo de 2 h, produzindo o mesmo efeito. Cada componente gradual já é individualmente invisível, e falhas compostas herdam essa propriedade.

O impacto prático é significativo. Os cenários C_1 e C_2 registram $P_d = 0{,}00$ com A entre $0{,}65$ e $0{,}87$, valores que seriam interpretados como operação saudável. No entanto, o erro acumulado de $+3{,}2\,°$C compromete decisões baseadas nesses dados, como o acionamento de climatização. A métrica Q, embora baixa, não captura a natureza sistemática do viés, reforçando a necessidade de análise conjunta. Essa vulnerabilidade motiva detectores não adaptativos ou com memória longa, como CUSUM, que acumula desvios sem redefinir a linha de base, ou filtros de Kalman [Delavernhe et al. 2025], que comparam a leitura com um modelo preditivo independente da janela móvel.

A abordagem proposta evidencia essa classe de vulnerabilidade ao reproduzir a degradação temporal de sensores em operação prolongada, cenário que testes unitários não exercitam por se restringirem a entradas instantâneas [Birchler et al. 2025]. Essa capacidade permite identificar precocemente limitações do detector antes da implantação em campo.

5. Discussão de Trabalhos Relacionados

Os resultados experimentais da Seção 4 podem ser situados na interseção de quatro linhas de investigação que, até o momento, evoluíram de forma independente. Esta seção discute trabalhos representativos de cada linha, reconhece suas contribuições e identifica a lacuna que a articulação entre elas permite preencher.

No âmbito da execução virtualizada de firmware, [Clements et al. 2020] substituem a camada de abstração de *hardware* por *handlers* de alto nível que permitem executar binários embarcados sobre QEMU sem *hardware* físico, enquanto [Feng et al. 2020]

geram automaticamente modelos de periféricos para habilitar *fuzzing* de firmware ARM. Ambas as ferramentas ampliaram a escalabilidade da análise de segurança, permitindo a detecção de vulnerabilidades de memória em firmwares comerciais. O método proposto compartilha a premissa de executar binários inalterados, porém direciona-a à avaliação de resiliência sob degradação sensorial. Enquanto o *fuzzing* emprega entradas aleatórias para maximizar cobertura de código, as campanhas realizadas aplicam modelos de perturbação determinísticos com dinâmica temporal contínua, o que permitiu revelar a vulnerabilidade do detector *z-score* a derivas graduais documentada na Seção 4.3.

No domínio da injeção de falhas por *software*, a ferramenta analítica de [Cotroneo et al. 2022] estabelece rigor estatístico para campanhas de injeção em sistemas de nuvem. A campanha aqui realizada adapta essa disciplina ao contexto IoT, onde os modos de falha relevantes não são computacionais, como *bit-flips* em memória, mas sensoriais, com dinâmicas temporais que exigem observação prolongada. O resultado $P_d = 0{,}00$ para derivas graduais corrobora experimentalmente a predição de [Erhan et al. 2021] sobre a dependência entre modo de falha e eficácia do detector, agora validada sobre firmware real em operação simulada de 24 h.

A viabilidade da co-simulação heterogênea como infraestrutura de teste para sistemas ciber-físicos, demonstrada por [Arrieta et al. 2019], motivou a sua adoção no método proposto. A distinção relevante reside no componente sob teste. Enquanto [Arrieta et al. 2019] exercitam modelos de *software* em linhas de produtos automotivos, a abordagem aqui proposta acopla o firmware de produção compilado via `native_sim` ao modelo ambiental por protocolo *lock-step*. Essa fidelidade permitiu confirmar que o *pipeline* do Golioth detecta descontinuidades em menos de um segundo ($T_d < 1$ s), conforme a Seção 4.2, comportamento que somente a execução do binário real poderia revelar.

Em uma quarta linha, focada diretamente nos algoritmos de detecção de falhas, esforços recentes exploram a modelagem preditiva em tempo de execução. Isso ocorre de forma incorporada via *TinyML on-device* [Ward et al. 2024] ou globalmente em nuvem através de Gêmeos Digitais [Jones and Connor 2025]. O escopo destas abordagens, contudo, é inerentemente a concepção de modelos de detecção. Esse foco algorítmico as desobriga de fornecer uma infraestrutura avaliativa sistemática capaz de mensurar a própria resiliência desses modelos frente a falhas físicas compostas e contínuas de longa duração.

A contribuição deste artigo reside, portanto, na articulação de elementos metodológicos até então isolados: a execução de firmware verdadeiro, o rigor de campanhas avaliativas e a co-simulação heterogênea. Essa integração viabiliza a injeção sistemática de falhas sensoriais em malha fechada. Desse modo, a plataforma transversal aqui proposta atende à carência de infraestrutura observada na área, entregando aos projetistas a capacidade de homologar e isolar as debilidades dos diversos algoritmos de detecção emergentes muito antes da sua implantação em campo.

6. Conclusão

Este artigo apresentou um método de avaliação de resiliência de firmwares IoT baseado em co-simulação heterogênea com injeção sistemática de falhas sensoriais. A campanha com o *Golioth Air Quality Monitor* revelou que a natureza adaptativa do detector de *z-score* móvel torna derivas graduais e travamentos completamente invisíveis ($P_d = 0{,}00$), enquanto descontinuidades abruptas são capturadas em menos de um segundo. Essa

assimetria persiste nos cenários compostos, onde componentes graduais mascaram a corrupção mesmo acompanhados de falhas detectáveis.

Esse achado só foi possível porque a co-simulação reproduz a degradação temporal de sensores em operação prolongada de 24 h, condição que testes unitários convencionais não exercitam por se restringirem a entradas instantâneas. A implicação prática é que firmwares IoT dependentes exclusivamente de detectores de janela deslizante estão potencialmente expostos a corrompimento silencioso sob modos de falha graduais e necessitam de camadas complementares com memória longa.

O escopo experimental adotado priorizou a profundidade da análise sobre um firmware representativo com modelo ambiental controlado e classificação manual dos resultados, decisão que viabilizou a identificação da vulnerabilidade estrutural reportada. Essa delimitação aponta naturalmente para três frentes de trabalhos futuros. A primeira consiste em incorporar perfis ambientais não estacionários derivados de séries reais de monitoramento. A segunda prevê a extensão da metodologia a firmwares adicionais e topologias multinó com falhas correlacionadas. A terceira investiga a incorporação de um oráculo automatizado baseado em classificador *fuzzy* para categorizar anomalias sem intervenção humana, viabilizando campanhas de larga escala.

Referências

Arrieta, A., Wang, S., Sagardui, G., and Etxeberria, L. (2019). Search-based test case prioritization for simulation-based testing of cyber-physical system product lines. *Journal of Systems and Software*, 149:1–34.

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Birchler, C., Khatiri, S., Rani, P., Kehrer, T., and Panichella, S. (2025). A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction. *ACM Transactions on Software Engineering and Methodology*, 34(5):152:1–152:9.

Clements, A. A., Gustafson, E., Scharnowski, T., Grosen, P., Fritz, D., Kruegel, C., Vigna, G., Bagchi, S., and Payer, M. (2020). HALucinator: Firmware re-hosting through abstraction layer emulation. In *Proc. 29th USENIX Security Symposium*, pages 1201–1218. USENIX Association.

Cotroneo, D., De Simone, L., Liguori, P., and Natella, R. (2022). Fault injection analytics: A novel approach to discover failure modes in cloud-computing systems. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1476–1491.

Delavernhe, F., Lecoeuche, S., and Karray, M. H. (2025). Kalman-based anomaly detection for IoT sensor streams in smart buildings. In *Proc. IEEE Int. Conf. on Industrial Informatics (INDIN)*, pages 1–6. IEEE.

Erhan, L., Ndubuaku, M. U., Di Mauro, M., Song, W., Chen, M., Fortino, G., Bagdasar, O., and Liotta, A. (2021). Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, 67:64–79.

Feng, B., Mera, A., and Lu, L. (2020). P2IM: Scalable and hardware-independent firmware testing via automatic peripheral interface modeling. In *Proc. 29th USENIX Security Symposium*, pages 1237–1254. USENIX Association.

Firouzi, F., Farahani, B. J., and Marinsek, A. (2022). The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). *Information Systems*, 107:101840.

Gaddam, A., Wilkin, T., Angelova, M., and Gaddam, J. (2020). Detecting sensor faults, anomalies and outliers in the Internet of Things: A survey on the challenges and solutions. *Electronics*, 9(3):511.

Golioth, Inc. (2026). Air quality monitor reference design. `https://github.com/golioth/reference-design-air-quality`. Firmware open-source sobre Zephyr RTOS. Acesso em: mar. 2026.

He, S., Shi, K., Liu, C., Guo, B., Chen, J., and Shi, Z. (2022). Collaborative sensing in Internet of Things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 24(3):1435–1474.

Isermann, R. (2005). Model-based fault-detection and diagnosis – status and applications. *Annual Reviews in Control*, 29(1):71–85.

Jones, M. and Connor, S. (2025). Digital twins for predictive maintenance in cyber-physical systems: A modern perspective. *ACM Transactions on Cyber-Physical Systems*, 9(2):15:1–15:28.

Minerva, R., Lee, G. M., and Crespi, N. (2020). Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, 108(10):1785–1824.

Natella, R., Cotroneo, D., and Madeira, H. (2016). Assessing dependability with software fault injection: A survey. *ACM Computing Surveys*, 48(3):44:1–44:55.

Ni, K., Ramanathan, N., Chehade, M. N. H., Balzano, L., Nair, S., Zahedi, S., Kohler, E., Pottie, G., Hansen, M., and Srivastava, M. (2009). Sensor network data fault types. *ACM Transactions on Sensor Networks*, 5(3):1–29.

Scilab Enterprises (2026). Scilab/xcos – open source software for numerical computation. `https://www.scilab.org/`. Acesso em: mar. 2026.

Ward, P., Smith, A., and Chen, D. (2024). Advances in TinyML: Empowering on-device intelligence for IoT sensorial resiliency. *IEEE Internet of Things Journal*, 11(4):2300–2315.

Zephyr Project (2026). Zephyr real-time operating system. `https://zephyrproject.org/`. Acesso em: maio 2026.